Concept of Interoperability

This chapter covers the following topics:

- Interoperability and EntireX Broker
- Messaging Model and Interoperability
- Communication Models and Interoperability

Note:
After viewing this chapter, see the chapter *Common Use Cases*, which supplies specific business examples of the interoperability available through EntireX Broker.

Interoperability and EntireX Broker

This section introduces the basic concept of EntireX Broker: achieving highly flexible interoperability of application components in a distributed processing environment. This concept is described from the perspectives of

- a messaging model
- communication models
- application programming interfaces
- EntireX components

in order to give you a comprehensive, high-level view of how EntireX Broker enables flexible interoperability between distributed application components.

Note:
Unless otherwise indicated, the communication model used in this section is client and server, and not publish and subscribe.

Messaging Model and Interoperability

Introduction

In a distributed processing environment that uses EntireX Broker, communication occurs through application components exchanging messages. An application component offering a service registers it with EntireX Broker (see `REGISTER`); this makes the service available to other application components able to communicate with EntireX Broker. An application component intending to access a service issues its request through EntireX Broker, which then routes the request to the specific application component offering the service.

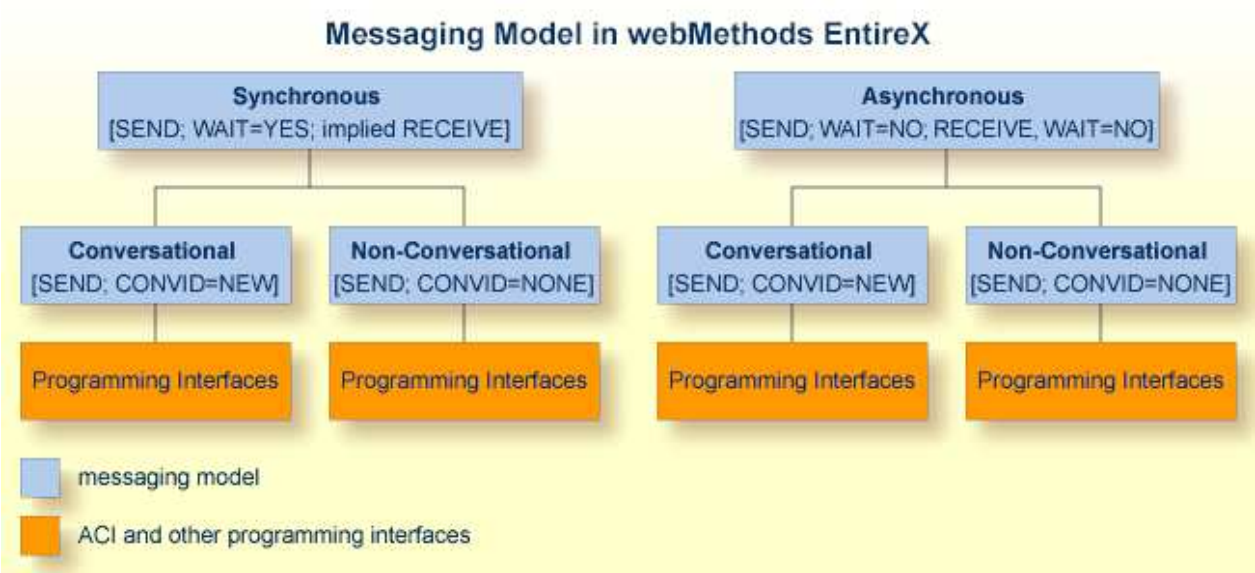
The following concepts help describe how message exchange is structured in EntireX Broker:

- **Synchronicity**
  The application initiating the request either waits for the result to return, whereby it suspends all processing (synchronous); or it does not wait for the result to return, whereby it is freed to do other processing (asynchronous).

- **Conversationality**
  The request can either be a single pair of messages comprising request/reply (non-conversational); or it can be a sequence of multiple messages which are all part of the same request (conversational).

Overview Diagram

The following diagram shows the two major concepts of EntireX Broker's messaging model: synchronicity and conversationality. See *ACI Syntax of Messaging Model* below for a description of the messaging syntax.



ACI Syntax of Messaging Model

The table below describes the messaging terms mentioned in the diagram above from the viewpoint of the application component initiating the request, as expressed in ACI syntax.

The ACI (Advanced Communication Interface) is the lowest level application programming interface that interacts with EntireX Broker. The ACI is common to all of the messaging models and communication models (see *Communication Models and Interoperability*) of EntireX.

Messaging Term		Client and Server		Publish and Subscribe	
		Client	**Server**	**Publish**	**Subscribe**
SYNCHRONICITY	Synchronous	• `SEND` (1) • `WAIT=YES` (1)	• `RECEIVE` • `WAIT=YES`	not applicable	not applicable
	Asynchronous (3)	• `SEND` • `WAIT=NO` • `WAIT=YES`	• `RECEIVE` • `WAIT=NO`	• `SEND_PUBLICATION` • `WAIT=NO`	• `RECEIVE_PUBLICATION` • `WAIT=NO` • `WAIT=YES` (2)
CONVERSATIONALITY	Conversational (3)	• `SEND` • `CONV-ID=NEW`	• `RECEIVE`	not applicable	not applicable
	Non-conversational (3)	• `SEND` • `CONV-ID=NONE`	• `RECEIVE`	• `SEND_PUBLICATION`	

Notes:

1. The synchronous `SEND, WAIT=YES` command contains an implied `RECEIVE` command.
2. The subscriber has the option of specifying `WAIT=YES`.
   Example: The subscriber uses a repeat loop that issues a `RECEIVE_PUBLICATION`. The advantage is that the program runs continuously, processing publications arising as random events, which simplifies programming effort.
3. Persistence available. See *Concepts of Persistent Messaging*.

Communication Models and Interoperability

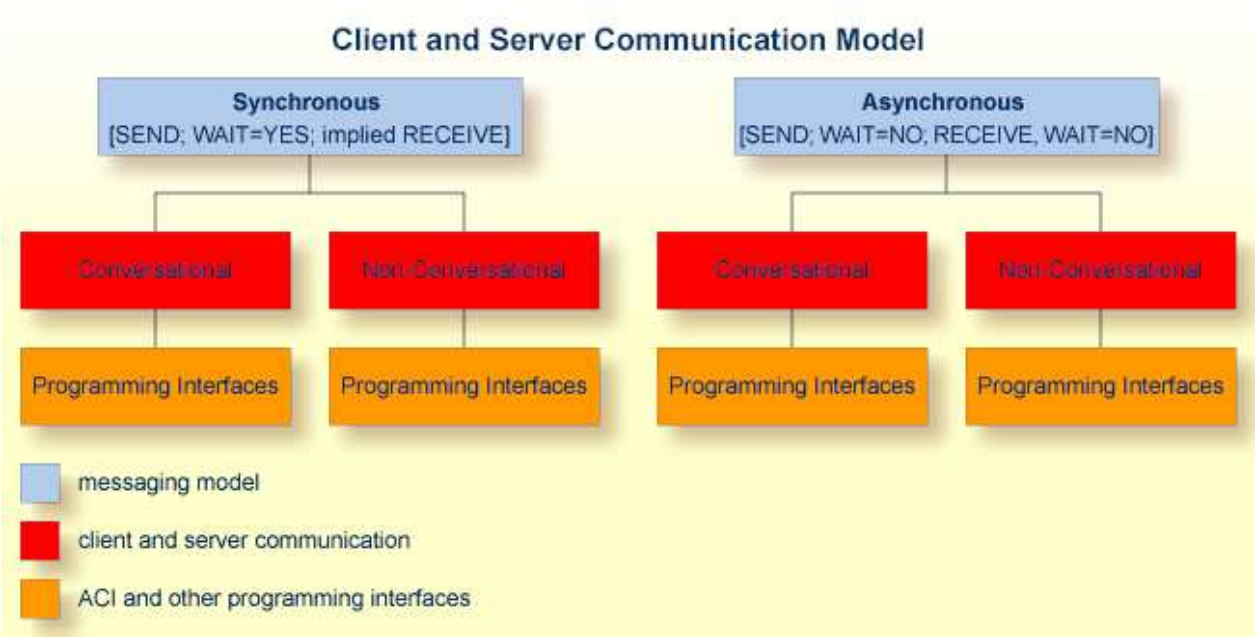
The EntireX Broker uses two communication models: client and server and publish and subscribe. Client-and-server communication is used if data is to be exchanged with exactly one partner; publish-and-subscribe communication is used if data is to be published. The ACI can be used for both client and server and publish and subscribe.

- **Client and Server**
  This model is based on the connection between exactly two partners: client and server. This model covers the requirements of conversational communication and asynchronous processing.

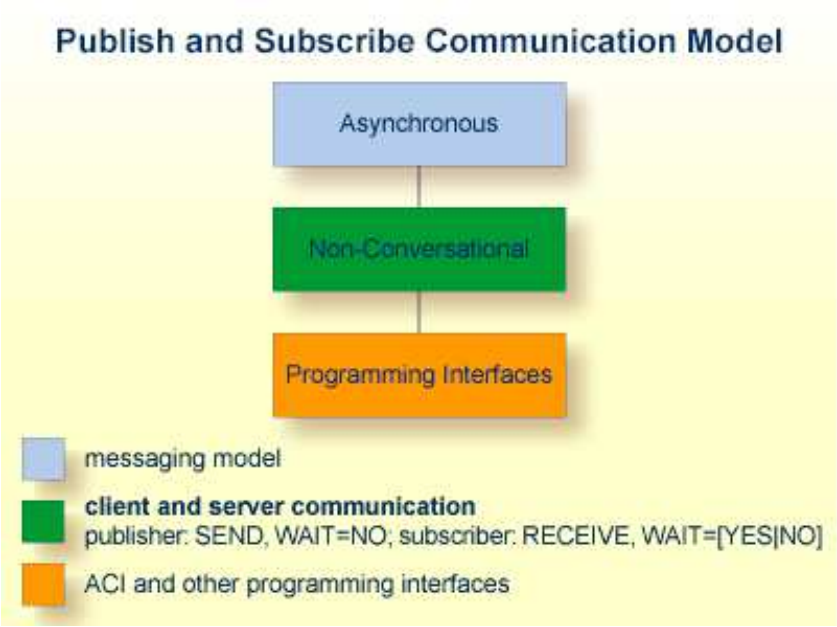
- **Publish and Subscribe**
  This model is implemented as an independent subsystem in the Broker, that is, an attribute determines whether it is set to active or inactive.

The following diagrams shows the two types of communication model used in EntireX Broker: client and server and publish and subscribe.

Client and Server



Publish and Subscribe



Publish and subscribe is normally classified as an asynchronous communication model. It is non-conversational in terms of message flow, that is, publications between publisher(s) and subscriber(s). The classification "asynchronous" is chosen because neither publisher nor subscriber directly depends on the activities of the other. The publisher always sends publications in a non-blocked manner.

Note:
The subscriber has the option of specifying `WAIT=YES` (see legend in above graphic). Example: The subscriber uses a repeat loop that issues a `RECEIVE PUBLICATION`. The advantage is that the program runs continuously, processing publications arising as random events, which simplifies programming effort.